[redacted]

From: [redacted]
Sent: Monday, May 08, 2006 1:10 PM
To: [redacted]
Subject: BOC9-2001-0011

Here is the disclosure.

Intellectual Property Law Department
[redacted]
8051 Congress Avenue, Internal Zip 4042
Boca Raton, FL 33487
e-mail address: [redacted]

----- Forwarded by [redacted] Boca Raton/IBM on 05/08/2006 01:02 PM -----

☒	**Disclosure BOC8-2001-0022** Prepared for and/or by an IBM Attorney - IBM Confidential Created By Greg Fitzpatrick On 03/13/2001 03:54:26 PM EST Last Modified By wpts1 wpts1 On 01/07/2005 08:24:24 PM EST Archived on 08/03/2002

Required fields are marked with the asterisk (*) and must be filled in to complete the form .

***Title of disclosure (in English)**
Distributed Storage of Granular Relation Elements from Within an RDBMS

Summary

	Status		Final Decision **(File)**
	Final deadline		
	Final deadline reason		
	Docket family		BOC9-2001-0011
*	Processing location		Boca Raton
*	Functional area		(11) Global Sales Operation & Technical Support [redacted] Div 91
	Attorney/Patent professional		[redacted]
	Invention development team (IDT)		[redacted]
	Submitted date		03/15/2001 06:39:10 PM EST
*	Owning division		SDG
	Incentive program		
	Lab		
*	Technology code		
	[redacted]		[redacted]

[redacted]

[redacted]

	Lab		
*	Technology code		
	[redacted]		[redacted]

Inventors with a Blue Pages entry

Inventors: Greg Fitzpatrick/Dallas/IBM, Jeffrey Heming/Dallas/IBM

Inventor Name	Inventor Serial	Div/Dept	Inventor Phone	Manager Name
> Fitzpatrick, G.P. (Greg) Heming, Jeffrey	[redacted]	[redacted]	[redacted]	[redacted]

> denotes primary contact

Inventors without a Blue Pages entry

Invention Development Team Information

To set the invention development team for a specific Functional Area and\or Technology Code, click on "Select...Functional Area" or "Select...TechnologyCode" in the action bar.

Attorney/Patent professional [redacted] team (IDT)

Response due to IP&L 04/16/2001

Main Idea

1. Describe your invention, stating the problem solved (if appropriate), and indicating the advantages of using the invention.

There are cases in which the data stored in a given RDBMS table is of such value (critical or sensitive) to an enterprise or governmental organization that it is unacceptable to have the content reside on a single physical machine (or single RDBMS instance). This may be unacceptable because of the threat posed by attack from an untrusted user (e.g., a hacker or simply an unauthorized user). The attack might come in the form of unauthorized access or simply by having a physical machine stolen from the premises. In these cases, enterprises would like to store this data in such a fashion that even physical removal of the machine on which the RDBMS resides does not give the perpetrator access to the sensitive data.

2. How does the invention solve the problem or achieve an advantage,(a description of "the invention", including figures inline as appropriate)?

Provided is a method for storing certain subsets of an enterprise's data at such a level of granularity on disparate machines that even in its raw (unencrypted) form the component pieces of the dataset provide no value to the perpetrator. Specifically, this invention provides an extension to RDBMS technology to allow entire tables, selected rows (relations) or selected columns of data to be selectively stored in a granular or "deconstructed" manner.

The invention provides a method whereby a database administrator may choose to have any subset of a table data stored in this "distributed" and "granular" fashion. Once the data subset has been selected (using any number of methods which constitute prior art) the invention is employed as follows: Rather than store the data on the same physical machine as the RDBMS, the data (and future updates, insertions & deletions) are stored on any number of other machines in a network of arbitrary size and complexity. Further, the elements of the data set may be stored down to a level of granularity chosen by the user. Within the RDBMS itself (potentially) no actual data is stored - only meta-data (similar to that in a stored procedure) consisting of instructions to reconstitute the disparate data.

For example, suppose an enterprise wants to store its personnel records using such a scheme. Given its penchant for risk aversion, the enterprise chooses to store the entirety of its employee table (assume it is a single table, for illustration purposes) in this manner. Further, assume that the enterprise wishes not only for each table row (relation) to be stored separately, but each column-entry for each row. In this example, if a row would logically consist of the elements FirstName, LastName, SSN, Band and Salary and the respective row values consisted of "Mark", "Palmer", "999-99-9999", "10", "100,000", then depending on the exact implementation algorithm, "Mark" could be stored on Machine A, "Palmer" on Machine B, "999-99-9999" on Machine C, "10" on Machine D and "100,000" on Machine E. Note that each of these elements may be stored either encrypted or in the clear.

(Again, this example is purely for illustration. In implementation, such a trivial machine assignment technique would never be employed. Rather, any number of sophisticated hashing algorithms might be an appropriate solution to this kind of problem.)

Returning to the example, the benefit of storing the elements of the row on a (non-predictable) set of machines in a non-trivial pattern causes the data to be much more secure, since access to a single machine does not allow the user to make the logical connection among the elements of the row.
Note that each column-entry is stored on a machine which may not have an RDBMS installed. In fact, the preferred implementation does not make use of an RDBMS on any target machine, although it use is not precluded. Rather, the "source" RDBMS contains the formula for finding the elements. The formula make take a number of different forms in implementation, but certainly must contain, for each element stored: machine unique ID (say, hostname or IP address), table unique ID, row unique ID and column name. The directive to store each element may take any of several common forms, including but not limited to fairly mundane ones like: FTP/HTTP get/post, standard RDBMS verbs like INSERT, DELETE, etc., or special verbs that could would call stored procedures from within the database to write the data and update the metadata following some cryptographic algorithm.

Each stored element might consist of nothing more complex than a flat file on the target machine's file system. Note that each stored element contains no trace of the source RDBMS or enterprise that created it, so discovery of disjointed elements would not point back to its source.

The invention contemplates further aspects of this core idea. For instance, data redundancy would be a prime concern in this example, since the failure of any given node on the "storage-plex" could cause large quantities of data to be corrupted. So, the invention further provides for multiple "writes" of any given data element to any number of machines. In our example above, the element "Mark" could be written, say, 4 times to any one of a pool of machines. Clearly, the formula stored in the RDBMS must contain sufficient meta-data to find any surviving elements of the redundant set.
In practice (say a disaster recovery scenario), only one instance of each element need be found to re-assemble the entire row.

The example above refers to column-entries beings stored (either encrypted or in the clear) as a unit.

[redacted]

However, this invention provides for even smaller elements to be stored - albeit at the increasing cost of *additional meta-data* per information unit. For example, suppose an enterprise cannot tolerate to have even the surname of a single employee revealed (think of the F.B.I., C.I.A. or N.S.A.). In that case, the user could choose to have no more than a single byte of a given column-entry stored as a unit. In an even more extreme example, single bits which compose the bytes in a given element could be the chosen storage element. Clearly, bits and even bytes of a given table row in isolation are generally of no use to a non-authorized user.

The most common implementation scenario would likely be to apply this invention to a pool of storage devices within a trusted zone, i.e., an enterprise's secure intranet/VPN, or possibly in the environs of a parallel database within a massively parallel processor (MPP) complex. However, if the units of data are sufficiently small and truly atomic, an enterprise could choose to make use of a pool of public storage devices - such as the large number of devices/services connected to the Internet. Essentially, any internet-connected device with read-write capability is a potential participant in such a virtual device pool. In this way, data could be stored in not just logically but physically separate locations - locations not even physically associated with the enterprise that owns the data. By utilizing this feature of the invention, along with sufficient redundancy and very granular data elements, an enterprise can achieve an arbitrarily high degree of data security, even though - paradoxically - pieces of those data elements reside outside the enterprise's trusted zone.

3. If the same advantage or problem has been identified by others (inside/outside IBM), how have those others solved it and does your solution differ and why is it better?
Traditional solutions to this problem have employed sophisticated password techniques and cryptography algorithms.
These techniques will almost certainly be used as adjuncts to this invention.

Distributed database technology requires instances of an RDBMS on every storage target machine and has failed to become predominant because of unwieldy maintenance demands of disparate non-enterprise systems. MPP systems have re-energized distributed concepts in a single enterprise manageable entity. The use of an MPP system with a parallel RDBMS is appropriate to the data security and availability characteristics envisioned by this invention.

4. If the invention is implemented in a product or prototype, include technical details, purpose, disclosure details to others and the date of that implementation.
N/A

[redacted]

[redacted] On what date was the invention workable? 03/12/2001 **Please format the date as MM/DD/YYYY** [redacted]
[redacted]
[redacted]

[redacted]

This [redacted] was entered by [redacted]/Boca Raton/IBM on 04/02/2001

What is the team's evaluation of this disclosure? Search
Date evaluated : 03/16/2001

[redacted]

[redacted] History	Who made the final evaluation	Final evaluation date
Search	[redacted]/Boca Raton/IBM	3/16/01

Search Information

Date sent: 04/02/2001	*Target completion date: 05/03/2001	Search results received date: 04/27/2001
Who was the search sent to [redacted]: WAIP;		
[redacted]		
[redacted]		

Search Office Information

Target completion date: 05/03/2001		Ship/Return date:
Search conducted by Kunkle		

[redacted]

Final Decision

This decision was entered by [redacted]**/Boca Raton/IBM** on **05/01/2001**

Decision: File	Status: N/A
PPM area:	
Date of final decision : 04/30/2001	

Additional filing information
Planned Filing date:
Filing comments:

Additional decision comments

Final Decision History

Entered on 1-May-2001 by [redacted]
File N/A 30-Apr-2001 Docket Family: BOC920010011

Post Disclosure Text & Drawings
To add additional information related to this disclosure once it has been submitted, click the action button below and a new document will be opened for you to enter the new information. To view existing post disclosure information, double-click on the item in the list below (if there has been additional information entered), and the document will open for you to view.

Form Revised (05/28/03)